Scripting "Junction Coal to Big Flats Power"

Part 2 - Converting the fully automated layout into an Interactive layout

This document is an explanation of the steps used to to convert our fully automated version of Eric Twickler's "Junction Coal to Big Flats Power" layout into an interactive layout. It is intended that these step by step notes may prove helpful to users who have developed a basic knowledge of scripting and are ready to design layouts that allow operator control of specific trains while running a fully automated background sequence.

If you followed the original article you will be aware that there are four Train Scripts on the automated version of this layout, plus a Master Script and seven Junction Action scripts.

Six of the Junction Actions are used to load and unload the trains, these will continue to fulfill this function and no changes to them are necessary. A small change will need to be made to the Junction Action Script at Big Flats to prevent it from issuing instructions to the operator driven train.

Three of the Train Scripts will continue to run their automated sequences, but minor adjustments will be needed to each of these to maintain a record of the positions of the trains, these changes are relatively straightforward.

We will also need to make several adjustments to the Master Script, these include defining some additional variables to record the train locations, and defining two "Procedures" to which we can make regular "CALL"s to check the state of the operator driven train, and to check the positions of the switches ahead of it.

These Procedures are in fact nothing other than user defined wait conditions, which hold up the running of the script until the operator has complied with his instructions. There is no need for the operator to acknowledge that he has completed a task, the script will check automatically to see whether or not he has done so.

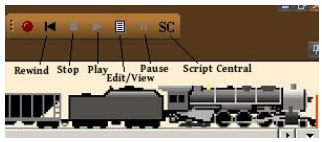
Step One - Prepare a new rrw file ready for editing

We start this project where the previous one ended by making a new copy of the scripted version of our layout.

To do this we should load the original, click the stop button on the Script Toolbar, select Save As from the file menu and save a copy of this file as **"junction_coal_interactive_s"**

We actually did this the easy way from Windows Explorer by copying and pasting **"junction_coal_to_big_flats_power_s"** and renaming the new copy as **"junction_coal_interactive_s"**.

Step Two - Making an extra copy of the Junction Action Script for Big Flats nested within Train Script number 1



We opened the new layout, and immediately clicked the STOP button on the Scripts Toolbar to stop the action, and then we clicked the Rewind button.

Then we opened Script Central (SC icon on the Scripts Toolbar), selected the "Junction Action" tab, selected the script for the Junction Action at Big Flats (JA7) and highlighted the whole of this text.

With the text highlighted we right clicked and selected Copy from the context menu.

Next we selected the Script Central "Scripts" tab, selected Train Script 1 (ID1) from the list, and scrolled down the text until we reached the Comment entry that refers to handing over control to the Junction Action at Big Flats. We placed our cursor just below this entry, right clicked and selected Paste from the context menu.

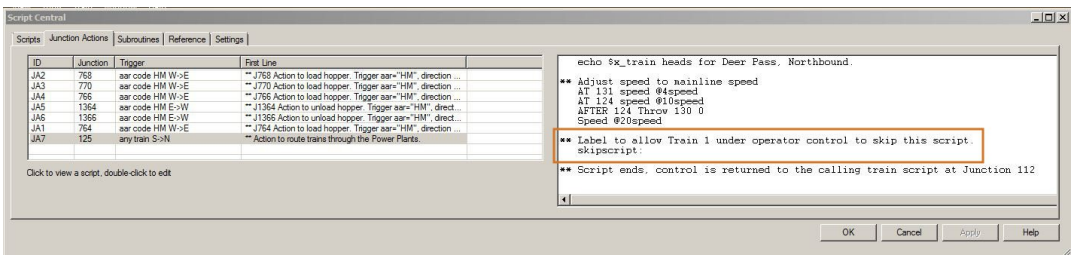
We now have all the script text relating to the train that we want to be controlled by the operator in a single script, Train Script 1 (ID1) ready for editing. But before we save our file again we first need to make a further adjustment to the Junction Action for Big Flats.

Step Three - Editing the Junction Action for Big Flats

We select the "Junction Action" tab in Script Central and click on the entry relating to the script at Big Flats (JA7).

We need to make an adjustment to this script to prevent it being executed when our operator driven train crosses this junction, this is because the commands for how that train behaves at Big Flats will be incorporated in Train Script 1.

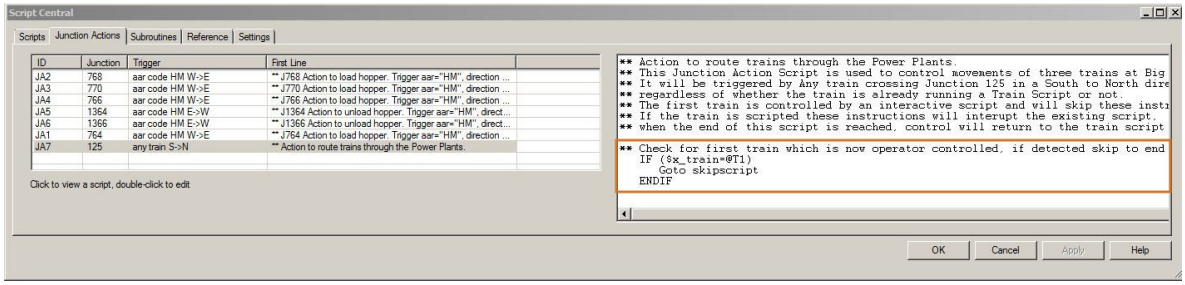
Fortunately making this script ignore our train is relatively easy to achieve, first we go to a line near the end of the script and add a Label. A label is a single word followed by a colon which serves as a marker in the script that we can direct the script to jump to, ignoring the code that precedes it. In this case we use the label: skipscript:



In this case it needs to be the very last executable command in the script and it is placed on the line immediately above the ** Script ends ..... comment.

Next we moved back to the start of this same Junction Action Script and added a conditional statement that causes the script to ignore all of its code IF (and only if) it detects that the train crossing the junction is the one we have converted to interactive. Hence the train name stored in T1 ignores this script, but the trains whose names are stored in T2, T3 and T4 will all load and execute this Junction Script.

This code goes immediately before the section to set the default route, if this train is detected the operator will have to set his own route as per his instructions in the Train Script. The automated settings will continue to function for the other three trains.



We need to save regularly by clicking Rewind Scripts on the Scripts Toolbar and selecting Save. Because this layout was set to Read Only in the previous exercise this defaults to the Save As dialogue with an incremental filename.

Step Four - Modifying the Note Window and references to it in the Master Script

Next we modified the size and position of the Note Window.

In the previous version this was an introductory note that quickly cleared itself. In this version it will be displaying regular instructions to the user.

We therefore needed a smaller Note panel that didn't obscure our view of the trains. The note window was therefore resized and moved to a position just below the Train Controller.

To do this we needed to temporarily disable the command to clear the note after three seconds by placing an asterisk (*) in front of it. We then resized and repositioned the Note Window.

We also repositioned the code in the Master Script for displaying the Note Window so that it appeared above the commands to Start the Trains. We moved this because if we left it at the bottom of the script the modified code in the interactive train script would have overwritten it.

Step Five - Additions to the Master Script

First we added some additional variable definitions which will be used to keep track of the locations of our four trains. These were not needed in our first scripted sequence because trains under total script control always behaved the same way when the script was run. In this version we are handing some of the control to the operator and it is unlikely that the operator's train will conform to the same identical timings as the scripted trains do.

By setting additional variables we can, for example, keep a record of when Train Block #1 is holding within the Deer Pass Siding, and when the opposing train Block #3 has cleared the switch to the Main. These additional variable definitions can be seen if you load the layout "Junction Coal Interactive_s" and study the complete Master Script in the Script Central dialogue.

We then needed to define a couple of Procedures within the Master Script that we could call regularly from the Train Script to check that the operator had followed the instructions given. These are essentially just additional Wait Conditions that are designed to prevent the script from running on until the operator has completed the assigned task.

The first of these was a procedure to check whether the train is moving or stopped, whether it is set to a forward or reverse direction, and to check how many cars are in the train.

The second was a procedure to check that instructions to throw switches have been complied with before allowing the script to continue. Once the required conditions are met the script is allowed to continue.

Our interactive Train Script will make numerous CALLs to these Procedures, each call will specify the required conditions for the script to proceed.

Remember, you can view the modified Master Script if you load the layout "Junction Coal Interactive_s" into TrainPlayer and open the Script Central dialogue.

The commented TPL code for both of our defined Procedures is also provided as an Appendix to this document.

Step Six - Modifying the Train Script for the first train to make it interactive.

Although this is the biggest part of the task, it is also the easiest. All that we needed to do was to work our way through the original script line by line and replace the automated commands with a series of prompts in the form of notes, with each prompt followed by a check to ensure the instructions have been fulfilled before we allow the script to move on.

For example, if we take the TPL code for moving our first train from its starting point through to the Scale we can compare the differences between the original automated script and the new interactive one.

Original Automated Script	New Interactive Script
** We want this train to start straight away, ** but the other trains will have a Wait Condition here ** Commands to Set route through Scale section Throw 138 1 Throw 142 0 Throw 148 0 Throw 168 0 Throw 167 0 ** Commands to Commence Loading Forward Speed @2speed echo @T1 loads Hoppers ** Hoppers will be loaded one by one by a Junction ** Action as they pass the Tipple	** We want this train started straight away, ** but the other trains will have a Wait Condition here NOTE Select Forward Gear CALL HoldUntil @T1 S F 12 NOTE Head out through Tipple\nSpeed limit is 5 mph CALL HoldUntil @T1 M F 12 NOTE Loading Hoppers\nSpeed limit 5 mph ** SET JCS to this Train SET JCS @T1 ** Hoppers will be loaded by a Junction Action as they pass the Tipple echo @T1 loads Hoppers ** Check for clear route as we move slowly forward CALL SwitchCheck @T1 138 1 A AT 138 CALL SwitchCheck @T1 142 0 A AT 142 CALL SwitchCheck @T1 148 0 A AT 149 NOTE Approaching Scale\nSpeed Limit 5 mph

The original automated script started by setting the route for the train, and only moving the train forward when the route was known to be clear. In the interactive version we want to simulate the brakeman walking ahead of his train through the yard and setting the switches accordingly. So we have reversed the order of these events and start the train moving first. If a switch ahead is correctly set the train will be allowed to move across it, but if it is not the train will need to stop until the brakeman sets the switch to its correct position.

The variable JCS is to record the fact that this train is occupying the Junction Coal Scale track, when the train eventually hooks up its Caboose this variable will be allocated to the next train to enable that train to start loading.

If we study the first instruction "Select Forward Gear", we will see that this is followed by a CALL to one of the Procedures we defined in our Master Script (Hold Until). The script passes four parameters to the Procedure.

1. The Train Name, extracted from the variable that stores it.
2. Whether the train needs to be Stationary or Moving (S or M)
3. Whether the train needs to be set to run in Forward or Reverse (F or R)
4. The number of cars in the train (for evaluating whether or not a required COUPLE or UNCOUPLE has been achieved).

The Procedure will then hold up any further processing of the Script until these conditions have all been met, in the case of this first instruction, only the third condition is outstanding and as soon as the user clicks the Forward/Reverse switch, control will return to the next instruction in the script which is to move forward (when a similar check is performed by a further CALL to the same procedure waiting for the operator to start moving the train).

The CALLs made to the other SwitchCheck Procedure work in similar manner, by passing the train name, the Switch Number, the position it needs to be in, and whether we are referring to a Switch ahead of or Behind the train. Only when the Switch is in the specified position will the Script be allowed to continue.

If you load the layout "Junction Coal Interactive_s" into TrainPlayer you can view the complete Train Script (Script 1) from within the Script Central dialogue window A small sample of the scripting commands used for the Meet at Deer Pass appears on the next page.

Step Seven - Modifying the other three Train Scripts

This was largely just a matter of adding code to set the location variables with the names of the trains entering them, and to set them to "clear" when the trains departed. Where two trains were operating in the same vicinity this information was then used to ensure safe train movements.

The original scripts were based on sequencing the trains and using the ON THROW wait condition by throwing this from another train when it was safe for the waiting train to proceed. This system is not suitable in an interactive situation because we can't be sure the operator will be at the right place at the right time. Which is why we changed to

recording occupancy of track segments for this version of the layout. This is not set up for Block Control as that degree of control is not needed in this situation which comprises a fixed sequence of only four trains. The way this actually works is best explained by comparing the actions of two trains as they make a Meet to cross each other at the Deer Pass siding.

Train Script 1 - The Interactive Script

```
** Approaching Deer Pass Northbound.
  AT 104
  NOTE Sound Horn\nSpeed limit 10 mph
  AT (7 113 30)
  NOTE Reduce Speed to 5 mph
  AT 112
  CALL SwitchCheck @T1 111 0 A

** Take the siding and wait for Southbound train to cross
  NOTE Enter siding slowly - Max Speed 5 mph
  CALL HoldUntil @T1 M F 13
  AT 111
  NOTE Prepare to Stop when\nConductor Signals Caboose is clear
  AFTER 129          <i.e. Northbound is now in the siding>
  NOTE STOP IMMEDIATELY\nConductor signals train is in the siding
  CALL HoldUntil @T1 S F 13
  CALL SwitchCheck @T1 111 1 B

** SET Block Occupancy Flags
  SET DPS @T1        <This clears the Southbound to go>
  SET DPBF "clear"
  NOTE Wait here for Southbound Train to Cross
  echo @T1 at Deer Pass, Northbound, awaiting Southbound train

** Wait until the Southbound train has cleared the North Switch
  While (DPM<>@T3); AFTER 0:0:01; endwhile
** <i.e. Southbound is in loop, North exit must be clear - let's go>
  CALL SwitchCheck @T1 108 0 A
  NOTE Sound Horn - Head out slowly
  CALL HoldUntil @T1 M F 13
  NOTE You must Stop for your Brakeman\nafter clearing the Junction
  AFTER 108
  SET DPS "clear"
  SET JCDP @T1
  CALL HoldUntil @T1 S F 13
  CALL SwitchCheck @T1 108 1 B
  AFTER 0:0:02
  NOTE Head out North for Junction Coal
  CALL HoldUntil @T1 M F 13
  AFTER 0:0:03
  NOTE Speed limit on downgrade is 30 mph
```

Train Script 3 - The Automated Script

```
** Approaching Deer Pass Southbound. Train will cross
** with Northbound train
  AT 110
  Speed @10speed
  Horn
  Throw 108 1
  AT 108
  Speed @4speed

** Release Northbound Train and continue Southbound
  AFTER 108          <i.e. Southbound is now in the Main Loop>
  echo @T3 at Deer Pass, Southbound.

** Train is clear of North Switch, set Block Occupancy Flags
  SET DPM @T3        <This clears the Northbound to go>
  SET JCDP "clear"

** Check to see if Northbound Train is wholly within Siding
** If it is already present proceed, if not Stop and wait.
  IF (DPS<>@T1)
    AFTER 0:0:02
    STOP
    ** Northbound is late, wait for it to arrive and stop in the siding)
    While (DPS<>@T1);AFTER 0:0:01; endwhile
 ** <i.e. Northbound is in Siding, South exit must be clear - let's go>
  Endif

** Continue South
  Throw 126 0
  Throw 111 1
  Forward
  Speed @4speed
  AFTER 111
  echo @T3 heads for Big Flats Power
  Speed @10speed
  AFTER 112
  Speed @20speed
```

A study of these actions will show that each train allocates its name to a variable once it is within the confines of the North and South switches. And that the other train tests these occupancy variables to ensure it is safe to proceed. The Southbound train will only STOP if the Northbound is not already confined within the siding, and its Brakeman has not reset the trailing switch.

Variations you might wish to try

That concludes the explanation of how the interactive script was added to Junction Coal to Big Flats Flower. I hope you find it useful. If you study the train positions you will see a Helper sitting there waiting to give the loaded train a push up to Deer Pass, why not try to add another script to call up the Helper.

To do this you would also need to adjust the timings of the following trains to give the Helper time to get back home to give them a push too. Apart from the timing changes to the Train Scripts, the Helper could probably be called up from a Junction Action script triggered by the departing train as it starts out after collecting a Caboose.

Richard Fletcher, April 2014

Appendix - The two procedures used for interactive testing of user reactions

The two procedures used in the Master Script for testing user response to the instructions are provided here for convenience. You can view the full Master Script, Train Scripts and Junction Actions from Script Central if you load the layout into TrainPlayer.

```
** DEFINE PROCEDURES

** HoldUntil, holds up the execution of a script until specific train conditions have been met.
** This can be used during user interaction to check if instructions have been complied with.

** Current version requires four parameters
** %1 Name of train for testing. (Can be passed literally, as contents of a variable, or as $x_train).
** %2 S if instructions require train to Stop, or M if instructions require train to start Moving again.
** %3 F or R, i.e. the direction setting needed on the Train Controller before allowing the script to
continue.
** %4 Train length (car count) needed before allowing script to continue, to check couple and uncouple
events.
**
PROC HoldUntil
  ** If train is not the required length, hold up the script.
  While ($train(%1,Ncars)<>%4); endwhile

  ** If train direction is not correctly set, hold up the script.
  While ($train(%1,Direction)<>%3); endwhile

  IF (%2="S")
    ** If train is required to stop, hold up script until it stops.
    While ($train(%1,Speed)>0); endwhile
  ELSE
    ** If train is required to move, hold up script until it starts moving.
    While ($train(%1,Speed)<1); endwhile
  ENDIF
ENDPROC

** End of procedure HoldUntil


** SwitchCheck checks the switch setting immediately ahead of (or behind) the train
** If the switch is wrong and the train is moving the driver will be instructed to Stop
** The Brakeman will then be instructed to throw the switch
** Requires four parameters
** %1 Name of Train, %2 Switch Number, %3 Required Position
** %4 A or B (is the switch to check Ahead of or Behind the train)

PROC SwitchCheck
  If ($Switch(%2)<>%3)
    If ($train(%1,Speed)>0)
      NOTE Stop before reaching the next switch\nand wait for your Brakeman to throw it
      While ($train(%1,Speed)>0); endwhile
    endif
    If (%4="A")
     NOTE Throw Switch Ahead of Train
    Else
     NOTE Throw Switch at Rear of Train
    endif
  ** Hold up script until the switch is thrown
  While ($Switch(%2)<>%3); endwhile
  NOTE Proceed with Caution
  endif
ENDPROC
```